DGPN-SCW-NOTES-March 2018

Rebel Cummings-Sauls, rebelcummingssauls@gmail.com
Boryana Koseva, boryana@ku.edu

Helpers: Ryan Otto, Kyle Hutson,Dave Turner, Carol Sevin, Adam Tygart

https://k-statecads.github.io/2018-03-20-gpn/

Day 1 Guest Book

Add your name here:

1. Rebel Cummings-Sauls
2. Behzad Ghanbarian (Mac, KSU Geology, ghanbarian@ksu.edu)
3. Konner Winkley (mac, KSU Biology, kmwinkley@ksu.edu)
4. Dechassa Duressa

5. Chad Bailey
6. Javier Fernandez (Windows 10, KSU Agronomy Department, jafernandez@ksu.edu)
7. Santiago Tamagno (Win10; Agronomy Department; stamagno@ksu.edu)
8. Prajaya Prajapati (Win 7; Agronomy, prajayap@ksu.edu)
9. Herman Coceancigh (Win, Chemistry, coceancigh@ksu.edu)
10. Luciana Nieto(mac/Windows| KSU Agronomy| lnieto@ksu.edu)
11. Teresa Shippy (Mac, Bioinformatics Specialist, KSU Biology; tshippy@ksu.edu)
12. Gowri Udayangani Kuda Singappulige (Windows 10, KSU Chemistry, gowri@ksu.edu)
13. Lan Lan (mac, KU Molecular Biosciences, lan@ku.edu)
14. Pratima Pandeya (Windows 10, KSU Chemistry, pratima@ksu.edu)
15. Zhoumeng Lin (Windows 7, KSU Anatomy and Physiology, zhoumeng@ksu.edu)
16. Taylor Simonson (Windows 10; KSU Dept. of Psych Sciences; tlsimons@ksu.edu)
17. Yi-Hsien Cheng
18. Scott Velasquez (Window; Educational Supportive Services)
19. Nelson Ramallo (Linux, Physics Dept., njramallo@ksu.edu)
20. Carol Sevin (Helper) (Windows 10; KSU Science Librarian; sevin@ksu.edu)
21. Wendy Reeves (Windows 10: biology)
22. TJ Wukitsch (Windows 10; KSU Dept. of Psych Sciences; wukitsch@ksu.edu)
23. King Henry VIII (Cobol)
24. Kyle Hutson (Helper) (Mac, K-State Beocat, kylehutson@ksu.edu)
25. Miao Li (Windows 10; KSU Dept. of Anatomy and Physiology; miaoli@ksu.edu)
26. Maverick Smith, (Windows 10; KSU Dept. of Psych Sciences, ms1434@ksu.edu)
27. Taliesin Hutson (Mac; KSU Computer Science; taliesin@ksu.edu)
28. Sidonia McKenzie (Windows; Dept. of Economics; sidoniam@ksu.edu)
29. Ryan Otto
30. Marvin Petingco(Windows; Biological and Agricultural Engineering; mpetingco@ksu.edu)
31. Yangfan Hao (Mac; yangfanh@ksu.edu)
32. Sandun Gajaweera(Windows 10:sandung@ksu.edu)
33. Adam Tygart (Mac, K-State Beocat)

Note: If you have a Beocat account, all of the software is already installed and you can login to Beocat and do all of the lessons from there.

folder name must not contain spaces (it can contain spaces. In order to read spaces "\ " must be used (e.g. for the folder "folder 1", you would use "cd folder\ 1"))

https://swcarpentry.github.io/shell-novice/reference.html

Session 1 - Shell (notes)

https://swcarpentry.github.io/shell-novice/01-intro/

The `--help` flag

Many bash commands, and programs that people have written that can be run from within bash, support a `--help` flag to display more information on how to use the command or program.

The --help flag doesn't work with Macs. Use man instead.

If you need to stop a process or the shell is hung, Ctrl-X will break out of it.

https://swcarpentry.github.io/shell-novice/02-filedir/

Options for commands can be specified with flags (a hyphen followed by character(s)) - these are case sensitive!

*ls -F	if there is trailing SLASH, it is directory
	* indicates executable file
ls	(means list)
ls -F	(case matters in Linux/cluster shells)
ls -f	(includes also the hidden files)
ls -Fl	(files/folders with details - dates, size)
ls -Flh	(files/folders with "human readable" - size in bytes)
pwd	(means "print working directory") - shows current directory
whoami	(gives username)
cd	(means change directory) - lets you move to a specified directory (e.g. cd Desktop)
cd ..	takes you to parent directory
cd	takes you to the last directory you were in
cd . or cd	[without any argument] takes you to home directory (~)
cd ../..	Up two levels
mkdir	make directory (never use spaces when making directories)
rmdir	remove directory. Only deletes empty directories
rm	remove file This is permanent, so be careful!
rm -r	removes a directory and everything in it. Be careful!!
head	displays first 10 lines of file on screen
tail	displays last 10 lines of file on screen
cat	displays entire file on screen - only good for small files
less	shows you the file one section at a time in a view (exit with q)
ls -lrth	list reverse human readable

Following a command with --help works for Windows but not Mac
Mac has to use man
man shows manual for a command
cp <file_name> <target_directory> copies file to a specified location
mv <file_name> <target directory> moves a file to a specified location

mv <file_name> <new_file_name> renames a file
wc shows line, word and character count for a file
    -l only line count
    -w only word count
    -c only character count
sort -n <file_name> sorts a file numerically and prints it to screen
uniq compares a line to the line above it and removes the line below if they are the same
    -c counts how many times the line occurred

File path - computer file system is like a tree
File path goes from root to branches
e.g. /home/user/folder/file
Relative path - tells the shell how to get to a certain place from where you currently are
Full path - unique path to a point, starting with root directory. Starting with / indicates full path.
Tab will autocomplete folder and file names when typing a file path.
* is a wildcard character that stands for zero or more characters and allows you to specify multiple files
You can combine the * with specific characters to get exactly the filenames you want
Enclosing characters in brackets [ ] requires that one of those characters be present at that position in the filename

> <file_name> directs the output of a command to the specified file
| is a pipe that allows you to connect commands. The output of the command before the | (pipe) will be used as input for the command after the pipe.

https://swcarpentry.github.io/shell-novice/03-create/
mkdir FolderName

nano is a good text editor for beginners
nano file_name will create a file and open the text editor
    Shows commands at bottom
    Ctrl-O saves the file
    Ctrl-X exits back to shell

“For loop” lets you iterate through a list of files and perform the same command on each file.

for filename in <files> (filename is a variable. You can call it anything - but best to use something that makes sense)
    do
    command (can have multiple lines)
    done
Shorthand format is...

for filename in N*[AB].txt; do head -n 3 $filename; done
$filename calls the variable “filename”

Good to first echo the command to make sure it does what you expect (e.g. echo “cp $filename original-$filename”). Then take out the quotes and echo.

Example use: making a backup copy of lots of files

Windows
cd /c/users/
ctrl-L to clear screen
Nano text editor installed with SWCarpentryInstaller.exe
https://github.com/swcarpentry/windows-installer/releases/download/v0.3/SWCarpentryInstaller.exe

cat - displays full file text.
less- progressive display
q - quits viewer
tail- last ten lines

Copying and moving Files
cp - copy
cp oldfilelocation newfilelocation

mv - move, also rename (mv old_name new_name)

After Break

https://swcarpentry.github.io/shell-novice/04-pipefilter/

- `command > file` redirects a command’s output to a file.
- `first | second` is a pipeline: the output of the first command is used as the input to the second.

https://swcarpentry.github.io/shell-novice/05-loop/

for someargument

```
do
        somecommand
done
```

Want to learn more about shell:
https://swcarpentry.github.io/shell-novice/06-script/
https://swcarpentry.github.io/shell-novice/07-find/
https://swcarpentry.github.io/shell-novice/reference.html
https://explainshell.com/

Full Course:
https://swcarpentry.github.io/shell-novice/

---Gowri

ls -F
ls -f
ls -F -a
cd - (last directory you were)
ls -Fa

rmdir directory_name (directory should be empty)
ls -Fl
ls -lh (human readable size) can also use ls -Flh
man ls (mac)
ls --help (windows)
head file_name (usually the first 10 lines)
tail(last 10 lines)
cat(whole content)
less (progressively show)

ls -lrth (ordered)
Question: ls -F is necessary in Mac? Just ‘ls’ would work in windows and linux
wc Word count (lists #lines #words #characters)
wc -l
wc -w
wc -c
sort -n file_name (n: numerically)
sort -r (reverse) or sort -nr (numerically reversed)
---pipes-- run multiple commands which uses the results of the commands
wc -l *pdb | sort (sort the output coming from the first command)

In animals /c/Users/Gowri/Desktop/Software-Carpentry/data-shell/data/
uniq file_name
uniq -c
sort salmon.txt >test | uniq
/Desktop/Software-Capentry/data-shell/north-pacific-gyre/2012-07-03
wc -l *[AB].txt (files ending with A.txt or B.txt)
for filename in N*[AB].txt
> do
> wc -l $filename
> done
Or you could type with semicolons
for filename in N*[AB].txt ; do echo $filename; head -n 3 $filename; done

USE echo and print the command before you actually run the for command
> (will replace)
>> (will append)

---Gowri

-Marvin-

ls -F if there is trailing SLASH, it is directory
ls means list
ls -F case matters in Linux/cluster shells
ls -f includes also the hidden files
ls -Fl files/folders with details - dates, size
ls -Flh files/folders with “human readable” - size in bytes
ls -lrth list reverse human readable
pwd means “print working directory” - shows current directory
whoami displays username
cd <folder_name> means change directory - lets you move to a specified directory
cd .. takes you to parent directory
cd takes you to the last directory you were in
cd . or cd [without any argument] takes you to home directory (~)
cd ../.. Up two levels
mkdir <folder_name> make directory (never use spaces when making directories)
rmdir <folder_name> remove directory. Only deletes empty directories
rm <file_name> remove file This is permanent, so be careful!
rm -r <folder_name> removes a directory and everything in it. Be careful!!
head <file_name> displays first 10 lines of file on screen
tail <file_name> displays last 10 lines of file on screen
cat <file_name> displays entire file on screen - only good for small files
less <file_name> shows you the file one section at a time in a view (exit with q)

wc <file_name>
wc *.<extension> e.g. wc *.pdb
wc p*.pdb display files that start with p and has a .pdb extension
wc *ethane.pdb display files that ends with ethane.pdb
wc -l *.pdb display lines of the files with .pdb extension
wc -w *.pdb display words of the files with .pdb extension
wc -c *.pdb
wc -l *.pdb > lengths.txt count the lines and save it to and make lengths.txt
sort -n
sort -nr reverse order
| pipe
wc -l *.pdb | sort -n
wc -l *.pdb | sort -n >sorted_lengths2.txt count lines, sort it, save to sorted_lengths.txt
uniq
wc -l *[AB].txt count lines of files with .txt extension that ends with either A or B
wc -l N*[AB].txt count lines of files with .txt extension that starts with "N" and ends with either A or B

For loop:

```
$ for filename in N*[AB].txt
> do
> wc -l $filename
> done
300 NENE01729A.txt
300 NENE01729B.txt
300 NENE01736A.txt
300 NENE01751A.txt
300 NENE01751B.txt
300 NENE01812A.txt
300 NENE01843A.txt
300 NENE01843B.txt
300 NENE01978A.txt
300 NENE01978B.txt
300 NENE02040A.txt
300 NENE02040B.txt
300 NENE02043A.txt
300 NENE02043B.txt
```

1st 3 lines in each file; foo is the given filename

```
$ for foo in N*[AB].txt
> do
> head -n 3 $foo
> done
1.03150932862
1.44755225695
0.224455411571
0.646620295293
0.529201331561
1.03149902331
2.16404354503
0.306827436349
3.64238088859
1.25588561078
1.17376037788
1.14263850494
0.974034913671
0.0395330178244
0.048747923634
0.142961371327
0.452337146655
0.332503761597
1.68204275762
0.788872109198
0.73714617485
0.490394468866
1.06438342275
0.121514392262
0.0929600007888
0.336622376971
0.216088857499
0.200595190005
1.16609257062
1.92486557634
1.8055719805
0.854421257637
0.1800430428
0.616254506154
0.283755587068
0.156990583983
3.68921390564
0.291020729177
0.869817390637
0.982953194226
0.231367385089
0.421897840681
```

```
$ for filename in N*[AB].txt; do echo $filename; head -n 3 $filename; done
```

```
NENE01729A.txt
1.03150932862
1.44755225695
0.224455411571
NENE01729B.txt
0.646620295293
0.529201331561
1.03149902331
NENE01736A.txt
2.16404354503
0.306827436349
3.64238088859
NENE01751A.txt
1.25588561078
1.17376037788
1.14263850494
NENE01751B.txt
0.974034913671
0.0395330178244
0.048747923634
```

Copy files and make it original-<filename>
For sanity check:

```
$ for filename in N*[AB].txt
> do
> echo "cp $filename original-$filename"
> done
cp NENE01729A.txt original-NENE01729A.txt
cp NENE01729B.txt original-NENE01729B.txt
cp NENE01736A.txt original-NENE01736A.txt
cp NENE01751A.txt original-NENE01751A.txt
cp NENE01751B.txt original-NENE01751B.txt
cp NENE01812A.txt original-NENE01812A.txt
cp NENE01843A.txt original-NENE01843A.txt
cp NENE01843B.txt original-NENE01843B.txt
cp NENE01978A.txt original-NENE01978A.txt
cp NENE01978B.txt original-NENE01978B.txt
cp NENE02040A.txt original-NENE02040A.txt
cp NENE02040B.txt original-NENE02040B.txt
cp NENE02043A.txt original-NENE02043A.txt
cp NENE02043B.txt original-NENE02043B.txt
```

```
$ for filename in N*[AB].txt; do cp $filename original-$filename; done
```

```
$ ls -F
goodiff                    NENE01971Z.txt           original-NENE01751A.txt
goostats                   NENE01978A.txt           original-NENE01751B.txt
Incomplete_NENE02018B.txt  NENE01978B.txt           original-NENE01812A.txt
NENE01729A.txt             NENE02040A.txt           original-NENE01843A.txt
NENE01729B.txt             NENE02040B.txt           original-NENE01843B.txt
NENE01736A.txt             NENE02040Z.txt           original-NENE01978A.txt
NENE01751A.txt             NENE02043A.txt           original-NENE01978B.txt
NENE01751B.txt             NENE02043B.txt           original-NENE02040A.txt
NENE01812A.txt             original-NENE01729A.txt  original-NENE02040B.txt
NENE01843A.txt             original-NENE01729B.txt  original-NENE02043A.txt
NENE01843B.txt             original-NENE01736A.txt  original-NENE02043B.txt
```

explainshell.com
Following a command with --help works for Windows but not Mac
e.g. ls --help
Mac has to use man
man shows manual for a command

mv <file_name> <folder>/
mv <file_name> <new_name>

File path - computer file system is like a tree
File path goes from root to branches
e.g. /home/user/folder/file
Relative path - tells the shell how to get to a certain place from where you currently are
Full path - unique path to a point, starting with root directory. Starting with / indicates full path.
Tab will autocomplete folder and file names when typing a file path.

https://explainshell.com/

-Marvin-

Feedback: pre-lunch day 1

Very good instructor
Positive: discover a new tool. Very helpful for big data sets.
Second half was very helpful for me, very happy. First half was about the things that I already knew. But it might have helped the people who are novice to it.
Learned at least one thing in each session. Learnt the meanings of what I have used and seen before but not really understood what those codes do.
First hour (8-9) was a bit of a waste for people who already had downloaded all software and data packages.
Pace after first hour was pretty good, given wide range of experience in attendees.
I was expecting more explosions.
Overall, this first half provided a lot of valuable information. The helpers were knowledgeable and make the experience a positive one overall.
Great hands-on examples
No coffee :(+2
I enjoyed everything. I thought it went well. I learned a lot!
Good intro into basics of bash, however I was hoping to get into shell scripts
For loops will be very helpful
Slow-moving for those who had experience - but I understand it was targeted at beginners.
I appreciated the depth of explanation, but it seemed to take quite a while to get beyond the most basic aspects

Great hanOn examples

Was helpful! I learned a lot. I understood that it was supposed to be pretty basic and not intended for more dvanced users.
I was lost at first but was able to catch up. And good cookies.
It was really helpful, but a little bit overwhelming.It
Helper were very great. Course is generally very useful. Instructors did good but bit fast catch up

Session 2 - MatLab/Octave (notes)

https://swcarpentry.github.io/matlab-novice-inflammation/setup/ to get the .zip for MatLab

Why use MatLab? Better UI than Python… but less powerful.
-- Matlab is used quite commonly both in industry and in education. They provide many ($$$) packages that can make your life very easy so you don't have to re-invent the wheel.
-- Also, Matlab can use the CPU very efficiently. Python can be made to be CPU-efficient, but it can be difficult to make it do so, especially when handling large arrays

```
csvread ('inflammation-01.csv')
```

Quotes force it to read the file as text rather than an array
Setting value of a variable

```
<variable_name> = <variable value>
```

; hides from displaying
[ ] multiple

```
disp(['Weight in pounds: ', num2str(weight_lb)]);
```

```
disp(['Weight in kg', num2str(weight_kg),char(10),'Weight in pounds: ', num2str(weight_lb)]);
```

M(:,6:end) → all rows from column 6 to the end (I don't know the last column #)

DEFENSIVE programming

- Checking the script during the run using display statements, etc.
- Help function_name or search
- Start small
- Simplify when possible
- Pre and post conditional tests

```
 for idx = 1:12
file_name = sprintf('inflammation-%d.csv', idx);
disp(file_name)
end

 for idx = 1:12
file_name = sprintf('inflammation-%02d.csv', idx); (two digits)
disp(file_name)
end
```

who - displays current variables
clear <variable_name> completely removes that variable
clear all - removes ALL variables
size <file or variable> outputs rows columns
class - gives data type

To pull a single value out of a table
If M = table
M(row,column)
: alone gives you all of one or the other
1:4 give you the range 1 through 4

-MATLAB notes - marvin
patient_data = csvread
('C:\Users\mpetingco\Documents\MATLAB\matlab-novice-inflammation-data\inflammation-01.csv')
disp or display
Semicolon don’t display output in command window

```
>> weight_kg=55;
>> disp(weight_kg);
    55

>> weight_lb=2.2*weight_kg

weight_lb =

  121.0000
```

```
>> display(['Weight in Pounds: ',num2str(weight_lb)]);
Weight in Pounds: 121
```

>> clear weight_lb clears array weight_lb

size gives number of rows and columns of array patient_data, respectively

```
>> size(patient_data)

ans =

    60    40
```

class gives the type of data (i.e single, double, etc)

```
>> class(patient_data)

ans =

double
```

Converting to integer

```
>> x=int16(325)

x =

    325
```

Generate an 8x8 array

```
M =

    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     1

>> M(5,1)

ans =

    32

>> M=magic(8)

M =

    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     1

>> M(5,:)

ans =

    32    34    35    29    28    38    39    25

>> M(1:4,:)

ans =

    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
```

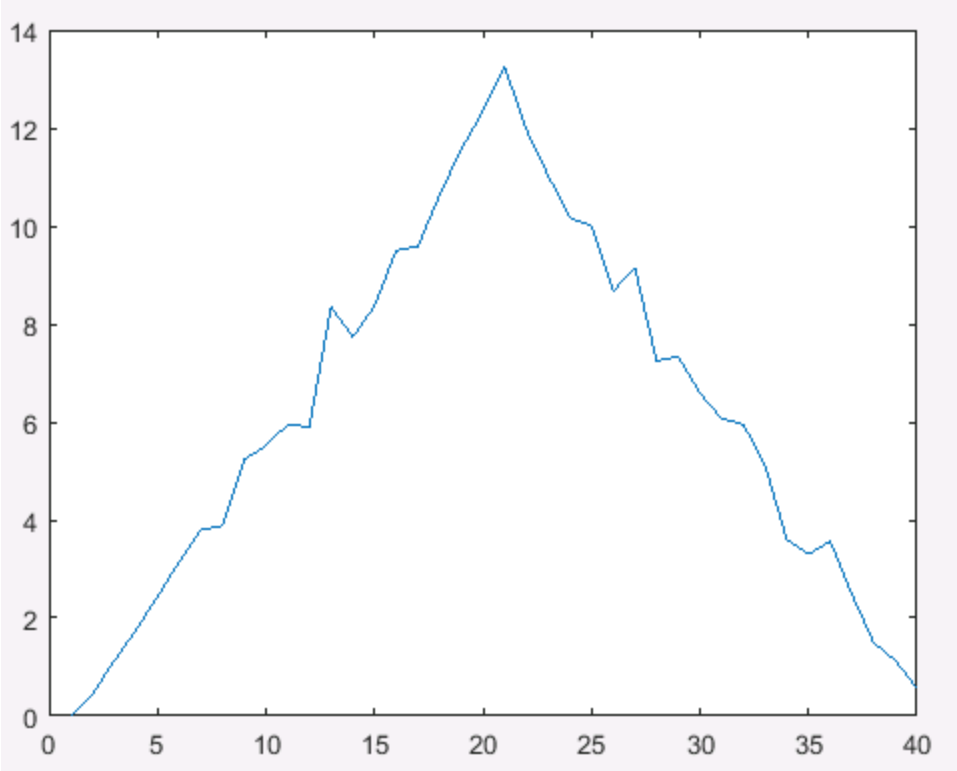
```
>> M(1:4,5:7)

ans =

    60     6     7
    13    51    50
    21    43    42
    36    30    31

>> M(2:3:end,:)

ans =

     9    55    54    12    13    51    50    16
    32    34    35    29    28    38    39    25
     8    58    59     5     4    62    63     1


M =

    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     1

>> M(2:3:end,2:3:end)

ans =

    55    13    16
    34    28    25
    58     4     1
```

```
M =

    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     1
```

```
>> disp(['Maximum of M:  ',num2str(max(M(:)))]);
Maximum of M:  64
```

```
>> disp(['Maximum of patient_data:  ',num2str(max(patient_data(:)))]);
Maximum of patient_data:  20
>> patient_data1=patient_data(1,:)

patient_data1 =

  Columns 1 through 20

    0     0     1     3     1     2     4     7     8     3     3     3    10     5     7     4     7     7    12    18

  Columns 21 through 40

    6    13    11    11     7     7     4     6     8     8     4     4     5     7     3     4     2     3     0     0

>> disp(['Maximum of patient_data1:  ',num2str(max(patient_data1(:)))]);
Maximum of patient_data1:  18
```

```
>> ave_inflammation=mean(patient_data,1);
>> plot(ave_inflammation);
```



-MATLAB notes - marvin

https://swcarpentry.github.io/matlab-novice-inflammation/02-scripts/

Feedback: end of day 1

Negative: It was hard to follow. Could not set the path correctly. Somehow managed to work in the directory
Problems with matlab/octave made this afternoon's session tough to follow. Good overview of matlab

The discussion of programming logic was good, but the speed was a bit rapid. It became a speed typing test towrads the end.

Hard to see the commands on the screen - also hard to keep up with speed. Would be good to tell the class about the online document with all the commands at the beginning of the session.

Glad to learn about Octave since it's free. +1

Speed was a bit quick. Definitely should say what you are typing out loud a few times.
I could not follow due to high speed.
I could not follow very well, maybe I need to look up the script more times.

Better to have larger font. Maybe need to include some introduction for the data set, it would help to better understand what the codes achieved.

It would be nice to discuss or make personal notes at the end of the day how we might practice in our own fields of study/work. Or during the sessions see if we can come up with examples (& not actually do them, just discuss).

I like the tip of group note-taking - my first time experiencing this. I didn't know this was an effective practice.

Day 2 Guest Book

SIGN IN

1.Rebel Cummings-Sauls
2.Konner Winkley
3.Wendy Reeves
4. Javier Fernandez
5. Kyle Hutson
6.Dechassa Duressa
7. Behzad Ghanbarian (Mac, KSU Geology, ghanbarian@ksu.edu)
8. Carol Sevin (Win10; KSU Science Librarian; sevin@ksu.edu)
9.Herman Coceancigh
10.Luciana Nieto (Mac|KSU Agronomy| lnieto@ksu.edu)
11. TJ Wukitsch
12. Pete Rosario
13. Lan Lan
14. Santiago Tamagno (Win10; Agronomy Department; stamagno@ksu.edu)
15. Zhoumeng Lin
16. Prajaya Prajapati (Win7, Mac ; Agronomy, prajayap@ksu.edu)
17. Yi-Hsien Cheng (Win 10; Anatomy and Physiology, yhcheng1987@ksu.edu)
18. King Henry VIII (Windows NT)
19. Taliesin Hutson
20. Miao Li (Win 10; Dept. of Anatomy and Physiology; miaoli@ksu.edu)
21.Scott Velasquez
22. Gowri Udayangani Kuda Singappulige (Windows 10, Chemistry, gowri@ksu.edu)
23. Chad Bailey
24.Maverick Smith (Win 10, KSU Department of Psych Sciences, ms1434@ksu.edu)
25.Pratima Pandeya (Windows 10, Department of Chemistry, KSU, pratima@ksu.edu)
26.Sandun Gajaweera(Windows 10, Dept. of Chemistry, KSU, sandung@ksu.edu)
27.Nelson Ramallo
28.Teresa Shippy (Mac, Bioinformatics Specialist, KSU Biology, tshippy@ksu.edu)
29.Laman Mamedova Mamedova@ksu.edu
30. Marvin Petingco(Windows; Biological and Agricultural Engineering; mpetingco@ksu.edu)
31. Adam Tygart
32. Taylor Simonson
33. Sidonia McKenzie

Session 3 - GitHub: Version Control

Setup - GitHub: Version Control

- Creating a repository
- Recording changes to files: `add`, `commit`, ...
- Viewing changes: `status`, `diff`, ...
- Ignoring files
- Working on the web: `clone`, `pull`, `push`, ...
- Resolving conflicts
- Open licenses
- Where to host work, and why
- Reference...

Git is a version control system that lets you track who made changes to what when and has options for easily updating a shared or public version of your code on github.com. You will need a supported web browser (current versions of Chrome, Firefox or Safari, or Internet Explorer version 9 or above).

You will need an account at github.com for parts of the Git lesson. Basic GitHub accounts are free. We encourage you to create a GitHub account if you don't have one already. Please consider what personal information you'd like to reveal. For example, you may want to review these instructions for keeping your email address private provided at GitHub.

Windows

Git should be installed on your computer as part of your Bash install (described above).

macOS

Video Tutorial
For OS X 10.9 and higher, install Git for Mac by downloading and running the most recent "mavericks" installer from this list. After installing Git, there will not be anything in your `/Applications` folder, as Git is a command line program. **For older versions of OS X (10.5-10.8)** use the most recent available installer labelled "snow-leopard" available here.

Linux

If Git is not already available on your machine you can try to install it via your distro's package manager. For Debian/Ubuntu run `sudo apt-get install git` and for Fedora run `sudo dnf install git`.

We'll do our work in the `Desktop` folder so make sure you change your working directory to it with:

```
$ cd
$ cd Desktop
```

Notes - GitHub: Version Control

http://swcarpentry.github.io/git-novice/

Git is a two stage process
File >>ADD>>staging>>>Commit

Configure Git in shell
git config --global user.name "<yourname>"
git config --global user.email "<youremail>"
git config --global color.ui "auto"
Windows: git config --global core.autocrlf true
OS X and Linux: git config --global core.autocrlf input
git config --global core.editor “nano -w”
git config --list
git init --help

mkdir <foldername>
cd <foldername>
git init
ls -a
git status

git add <filename>
git status

git commit -m “<file description, changes made, etc.>”
git log

git diff

git diff --staged
git commit --amend ‘used to edit latest commit
git log -1 ‘Returns last commit details
git log --oneline ‘Returns one line description
git add <filename> <filename> ‘to add multiple files

git diff HEAD <filename> ‘Returns difference between current non-committed version and last committed version

git diff HEAD~1 <filename>

```
echo “message” > mars.txt  ‘Replaces contents of file
echo “message” >> mars.txt  ‘Appends to file

git checkout HEAD mars.txt  ‘Restore most recent version from repository

.gitignore
results/                ‘will ignore this directory
!results/final.dat      ‘will not ignore this file
```

Feedback: pre-lunch day 2

Please add one positive and one negative from the morning lesson and instruction.

This was very useful and the pace was very good for me. Thank you!
This was a really nice in-depth experience. The concepts seemed to move a little slowly at times, and it got a little repetitive at times. It would have been nice to move through some of the material a little more quickly.
Very helpful
Great job. Many steps but well worth it.
Very helpful
First time user…. The instructions were clear and easy to follow… facilitators are extremely helpful… I enjoyed this session. Maybe the instructor could provide more examples for applications to research writing and compatibility with other applications
Pace was good, make the files more interesting!
It was great! Exactly what I need. I make a lot of changes for my scripts and get confused when I come back and use the script again. I need a lot more practice though. This was amazing.
Just enough to see the utility and power of Git and the need for practice!

You are a great teacher.
Good pace for feeling confident. Git will be a very useful tool.

Git was awesome! Great with the explanations!

Git will be very useful in tracking revisions in scripts. Maybe some example using matlab script and tracking the revision that have been made. Thanks.

Git is really cool! Very useful workshop!

Fast lecture pace too fast but content could be very helpful for coders.

After Lunch

Session 4 - Programming in R

Setup - Programming in R

- Working with vectors and data frames
- Reading and plotting data
- Creating and using functions

- Loops and conditionals
- Using R from the command line
- Reference...

This lesson assumes you have the R, RStudio software installed on your computer.

R can be downloaded here.

RStudio is an environment for developing using R. It can be downloaded here. You will need the Desktop version for your computer.

You also need to download some files to follow this lesson:

1. Make a new folder in your Desktop called `r-novice-inflammation`.
2. Download r-novice-inflammation-data.zip and move the file to this folder.
3. If it's not unzipped yet, double-click on it to unzip it. You should end up with a new folder called `data`.
4. You can access this folder from the Unix shell with:

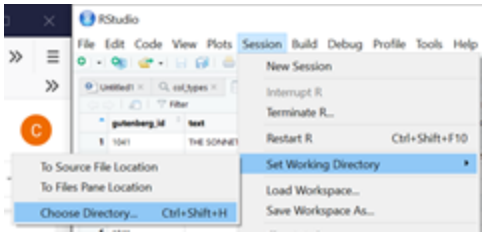
```
$ cd
$ cd Desktop/r-novice-inflammation/data
```

Notes - Programming in R

http://swcarpentry.github.io/r-novice-inflammation/

Data set is data on inflammation in patients being treated with a new arthritis drug
Comma separated value (csv) files
Rows are patients
Columns are days

Set working directory
Session
    Set working directory
    Navigate to folder
    Open
That creates this command, which you could also type in to the Console
setwd("~/Desktop/r-novice_inflammation")
    Note: Windows users have to type the complete path or session>>set working directory>>choose directory



getwd() get working directory
read.csv() import a csv file (file="<filename>", header=TRUE/FALSE

Tip: use tab to auto-fill
class() determine the class
dim() size/dimensions
view() print/show data
= assigns
<- assigns

Looking at parts of a spreadsheet
Assign file to a variable
<- is called the assigner
dat <- read.csv(file="data/inflammation-01.csv", header = FALSE)

dat[1,2] gives the value of the first row, second column
1:5 gives 1 through 5
1,3,5 gives 1, 3 and 5
Leave the spot blank to get whole row or whole column

Can also use column and row names in same format IF you set header to TRUE

Can combine this with functions to run a function on particular subsets of data
For example,

min(dat[, 7]) will give the minimum value in column 7

Use as.numeric to convert all to numbers (columns are defined as vectors so they will always work with numeric functions, rows need as.numeric before min/max/median/etc)

summary(dat[,1:4]) calculates a summary: the minimum value, the first quartile, the median, the mean, the third quartile and the max value.

Using the apply function
Example:
avg_patient_inflammation <- apply(dat, 1, mean)
    In this case 1 = Rows and 2 = Columns

```
analyze <- function(filename) {
  # Plots the average, min, and max inflammation over time.
  # Input is character string of a csv file.
  dat <- read.csv(file = filename, header = FALSE)
  avg_day_inflammation <- apply(dat, 2, mean)
  plot(avg_day_inflammation)
  max_day_inflammation <- apply(dat, 2, max)
```

```
  plot(max_day_inflammation)
  min_day_inflammation <- apply(dat, 2, min)
  plot(min_day_inflammation)
}
```

Changing from csv to pdf = sub("csv", "pdf", f)

Feedback: End Day 2

Examples are helpful. I have not used R before and this workshop is a good introduction of what I can do with R. Git can also be useful in tracking revisions with my programming projects.

Overall, I don't feel like I have gotten that much from the R portion of the workshop. It was hard to follow due to: 1) the lack of description within long lines of script, 2) lack of fluency in the presentation, 3) rifts that were occuring between separate instructors during this portion of the workshop.

More context when we're using the examples - the source of the inflammation data was only mentioned at the beginning of the R session.

Went a little too fast for me. I need a whole day for R.

Great job Rebel.

Very informative but it went a little too fast in the end. All the very best in your new endeavors Rebel! You did a great job! :) Check in more with the audience from time to time.

Like you said, the content of the presentation became a little deep for an introduction. Plus the eye was more focused on the clock. :) Great job.

It's really helpful, thank you Rebel. This workshop gives a different perspective of using R studio. Best wishes to your new life.

You covered very important concepts in R. Too fast paced to catch up typing commands
Great job.
Thank you for great workshop

This was an excellent workshop. Thank you guys for your time and effort on this. Definitely, a very useful course. Thank you!

It was a good introduction to R and gave me enough of a foundation to feel comfortable learning more on my own.

It helped me a lot.

Thank you for this valuable workshop.

I really think this workshop is a great idea. It is true that sometimes it becomes slower or without flow, but i get it. It would be nice to have the same workshop only for R or matlab, or maybe a second level to keep working. However, thanks a lot for the time.